Page 88. Please amend the Abstract as follows.

A method and apparatus for implementing a shared message queue using a list structure. A put list is defined comprising a sequence of list entries, each of which corresponds to a message in the queue and has an associated list entry key. Each list entry key corresponding to an uncommitted message falls within an uncommitted key range defining an uncommitted portion of the put list, while each list entry key corresponding to a committed message falls within a committed key range defining a committed portion of the put list. To write a message to the queue, a list entry is added to the put list having a list entry key within the uncommitted key range. ~~List entry keys in the uncommitted key range are assigned in order of message priority and in order of arrival time for messages of a given priority. Each list entry key has a more significant portion indicating the list portion to which the corresponding list entry belongs and a less significant portion indicating the order of the list entry in the list portion. To commit messages to the queue, the list entry keys associated with the list entries are modified to fall within the committed key range to move the list entries to the committed portion of the put list while preserving their relative order as determined by the list entry keys. To read a message from the queue, the list entry whose list entry key has the lowest value in the committed key range is moved from the committed portion of the put list to a get list defined for the queue. If there are no messages currently in the committed portion of the put list, the requester waits and is notified of a subsequent change in state of the committed portion from an empty state to a not empty state. To abort a read of the message from the queue, the list entry is moved back from the get list to the committed portion of the put list. To commit a read of the message from the queue, the list entry is removed from the get list.~~

A3